POSITION PAPER

ON

UNIFIED MODELING LANGUAGE

1. The Department of Defense (DoD) software acquisition and development professionals should embrace the Unified Modeling Language (UML) standard for modeling software systems. The UML is a predominantly graphical notation that software development methods use to express software systems' designs (Fowler 2000). Popkin Software provides an astute definition of the UML:

> "The Unified Modeling Language prescribes a standard set of diagrams and notations for modeling object-oriented systems, and describes the underlying semantics of what these diagrams and symbols mean."

Basically, the UML provides a standardized visual tool for designing and describing software systems. This standardized visual tool is quite helpful in communicating system requirements and has been widely adopted by commercial software development companies. Even with its extensive popularity, there exist a few opponents to UML who contend that the language is complex, requires costly training, and is difficult for tool vendors to implement. These obstacles are small compared to the benefits of implementing the UML in all future DoD software-intensive projects.

2. The UML provides a standardized visual tool for designing and describing a software system's architecture. Air Force Software Technology Support Center (STSC) exhibits the immense importance of a software system's architecture by concluding that the architecture must be articulated, include provisions for change, and must be controlled and maintained throughout

the system's lifecycle (STSC 1996). The STSC provides insight into the process of creating and maintaining the system architecture, yet does not provide guidance on what tools to use to describe an architecture. This is where the UML becomes invaluable. Created by well-known methodologists, Grady Booch, Ivar Jacobsen, and James Rumbaugh, one of the primary design goals of the UML was to "provide a standard language for visualizing, specifying, modeling, and documenting the architecture of a system" (Melewski 1998). Two key words "standard" and "visual" are what set the UML apart from its modeling language predecessors. In the past, to understand a software system's architecture, one would have to read through the system's code or use non-standard design notation. The Object Management Group (OMG), a consortium of software vendors founded in April 1989 and committed to developing technically excellent, commercially viable and vendor independent specifications for the software industry, has recently standardized the UML version 1.3 (http://www.omg.org/omg/background.html). So now, like in so many other engineering and architectural disciplines, the software engineer has standard tools to express how a software system is to be built. Much like a blueprint, any person who has the knowledge to read the standardized notation can understand what the system designer intended. With the UML there is no need to argue over how to express the design, enabling system designers to work on the actual design (Popkin 1998). As proven in other engineering disciplines, significant economies can be achieved by agreeing on a common terminology (Melewski 1997). Standardization also strengthens the communication between the people involved in software development projects (Fowler 1998).

3. Lack of communication has been cited as one of the main reasons errors occur during requirements definition and analysis (STSC 1996). Requirements errors compose over 50% of all software errors according to data collected from Rome Laboratory (STSC 1996). According

to Jim Van Buren, technical program manager at the STSC, “Requirements management is the requirements issue that most impacts military software projects. Capers Jones found that 70 percent of all military software projects are at programmatic risk because of requirements volatility.” The UML is invaluable in communicating software system requirements; its robust notation is especially helpful in handling requirements design (Melewski 1997). Communicating system requirements is important between the user and the system designer and between the system designer and developers who are not involved in the analysis phase (Melewski 1998). Martin Fowler, a UML expert and author of two editions of the best-selling book *UML Distilled*, believes the key to developing good software is skillful communication and a thorough understanding of the users’ world (Fowler 2000). The UML provides important communication tools to help software developers communicate (Fowler 1998). One such tool, called a Use Case, provides the basis of communication between customers and developers in planning the project (Fowler 2000). A Use Case is a ‘snapshot’ of one aspect of how a software system is used by a person. For example, the spell-checking feature in a word processing product would constitute one Use Case. Combining all the Use Cases should produce a complete representation of the external requirements of the software system. The Use Case diagram provides the entry point into software system requirements analysis; other UML tools are used to show system sequences, activities, states, associations, attributes, and interfaces. Formal Software Requirements Specification (SRS) documents usually exist or are developed for DoD software projects. Each SRS requirement should be fulfilled by a correlated Use Case. If a requirement has been overlooked, Use Case modeling enables the discovery of requirements; conversely, Use Cases can be employed to validate a system against its initial requirements (Popkin 1998). Bill Richards, software product manager at Hilco Technologies Inc., St. Louis has found the UML invaluable in requirements design, and also in gaining customers consensus on requirements (Melewski 1997). This leaves little to wonder when considering why the UML has become so popular in such a short amount of time.

4. The UML has been widely adopted in software development companies and became a de facto standard before it was standardized by the OMG (Melewski 1997). One reason for the wide acceptance of the UML is that it is the unification of the methodologies of the so-call "three amigos" -- an industry nickname given to the UML authors Booch, Rumbaugh, and Jacobsen. According to Deborah Melewski, the UML "results from collaboration of large and small software suppliers and consultants, including Hewlett-Packard Co., IBM, Icon Computing, i-Logix, IntelliCorp, MCI Systemhouse, Microsoft Corp., ObjectTime Ltd., Oracle Corp., Ptech, Platinum Technology, Rational Software Corp., Sterling Software Inc., Taskon and Unisys Corp" (Melewski 1997). In addition to co-submitters of the UML standardization proposal, many software developers became UML endorsers (Melewski 1997). Many companies have been utilizing UML for since its inception. Martin Fowler claims "the UML has become the standard way to draw diagrams of object-oriented designs, and it has also spread to non-OO fields." He continues by proclaiming "pre-UML methods have all died out" (Fowler 2000). Paul Evitts, another UML expert, agrees that the UML has widespread support in the object tools industry (Evitts, April 1998). Even with its widespread popularity, unanimous agreement on any topic in software development is about as likely as a unanimous agreement in Congress.

5. Opponents to UML contend that the language is complex and lost some of the initial plan's simplicity because of its authors' lack of consensus (Melewski 1997). It has been suggested that the co-submitters solved disputes by including everything into the specification (Melewski 1997). Often compared to the complexity of the English language, UML's

ambiguities, size and complexity arouse the most concern from those trying to learn the modeling language (Melewski 1998). Since the UML is more sophisticated than earlier notations it requires some study to understand (Evitts, Feb 1998), therefore, implementing the UML can be a learning challenge (Melewski 1998).

6. As with any new standard, learning the UML will require costly training. The implementation costs of the UML can be a corporate issue (Melewski 1997). Yet, local training firms and UML training in universities could alleviate some UML implementation costs (Melewski 1997). There are numerous Internet sites that provide free insight and tutorials on the UML. A myriad of books, articles, and trade magazines are available to educate and inform. The Air Force Institute of Technology (AFIT) provides a distance learning course, CSE 494 'Object-Oriented Analysis and Design', which reviews the UML. The Software Program Managers Network (SPMN), directed by Congress to train U.S. personnel with the required skills to design and build large-scale software systems (http://www.spmn.com), could add UML courses to their 'Software Best Practices' curriculum. There would be a high return on investment (ROI) of the training costs considering the savings brought about by efficient software development and reuse.

7. Opponents to UML argue that the UML specification is virtually impossible for tool vendors to implement completely (Melewski 1997). Yet, the UML is designed so that each vendor can implement different levels of compliance based on what is needed and can decide how much of the UML to implement (Melewski 1997). Numerous vendors, including Air Force preferred Microsoft Corporation, Visio Corporation, and Rational Software Corporation, provide UML diagram support products. UML tools vendors are likely to provide incremental updates to their products to eventually support the full UML specification (Melewski 1998).

8. In my opinion, use of the UML in the DoD should receive more focus since the benefits of UML use far outweigh the implementation costs and headaches. The UML has been widely accepted and implemented by a large majority of software development firms. These firms will be using the UML for modeling our software-intensive systems in the future, affecting our future software acquisitions. According to the STSC:

> "To understand the acquisition reform taking place in DoD, one must understand that its fundamental purpose is to enhance and unify the commercial and defense industrial base by applying the most modern industrial products, processes, and practices to our acquisitions. For software, this includes adapting the most modern methods and principles of software engineering."

9. The UML is clearly a modern software engineering method which enables standard software architectural design, making implementation, maintenance, and reuse easier, while avoiding data redundancy (Bereny 1999). This leads to saving in both time and money, considerably increasing the efficiency of software development teams (Bereny 1999). The software development industry has an 80% to 90% failure rate at large-scale, mission-critical projects (Jones 1996) – often called the 'software crisis.' While I don't believe the UML is a 'silver bullet' for the software crisis, I do believe it is a step in the right direction, much like the Software Engineering Institute's Capability Maturity Models. Modern UML tools, such as Visual Modeler in Microsoft's Visual Studio 6.0 Enterprise Edition, permit automatic code generation– further increasing the efficiency of software development teams. Touting the UML's flexibility, extensions to the UML provide tools to create specialized models for specific domains such as business processes and real-time systems (Kuncel 1999).

10. The Department of Defense (DoD) software acquisition and development professionals should embrace the Unified Modeling Language (UML) standard for modeling software systems. DoD support and promotion of the UML would help to mitigate the software crisis via thorough requirements analysis and standardized documentation and communication methods. The potential savings of billions of taxpayer dollars and innumerable person-months of software development time is certainly worth the effort of adopting the Unified Modeling Language.

References

Bereny, Naveena, “Data Modeling with Rational Rose”, *Rose Architect Magazine*, Rose 101 Copyright 1999 Miller Freeman Inc., a United News & Media Company.

Evitts, Paul and Hichcliffe, Dion, “UML: The Final Stretch”, *Object News – UML Opinion Page*, http://www.objectnews.com, February 17, 1998.

Evitts, Paul and Hichcliffe, Dion, “Reaction to OVUM UML Assessment”, *Object News – UML Opinion Page*, http://www.objectnews.com, April 21, 1998.

Fowler, Martin, *UML Distilled*, Second edition, Addison Wesley Logman, Inc., 2000, pp. xvii, 1, 10, 18.

Fowler, Martin, “Why Use the UML?”, *Software Development Online DesignCenter,* http://www.sdmagazine.com/uml/, October 1998.

Jones, Capers, *Pattern of Software Systems Failures and Success,* International Thompson Computer Press, 1996.

Kuncel, Joseph S., “Next Millenium Technologies – Experiences Of An Early Adopter”, Software Technology Conference White Paper, May 1999.

Melewski, Deborah, “Ready for Prime Time?”, *Application Development Trends*, November 1997.

Melewski, Deborah, “UML Gains Ground”, *Application Development Trends*, October 1998, pp. 34-44.

Popkin Software, ”Modeling Systems with UML”, *Popkin Software White Paper*, Copyright 1998 Popkin Software and Systems.

Software Technology Support Center (STSC), *Guidelines for Successful Acquisition and Management of Software-Intensive Systems*, June 1996, pp.1-26, 2-19, G-20.

Van Buren, Jim, and Cook, David A., “Experiences in the Adoption of Requirements Engineering Technologies” *Crosstalk The Journal of Defense Software Engineering*, December 1998, pp. 3-10.